METHOD, SYSTEM, AND PROGRAM FOR PROCESSING TRANSACTION REQUESTS DURING A PENDENCY OF A DELAYED READ REQUEST

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] The present invention relates to a method, system, and program for processing transaction requests during a pendency of a delayed read request.

2. Description of the Related Art

[0002] The Peripheral Component Interconnect (PCI) bus architecture provides a low latency path through which devices implementing the PCI architecture can communicate. Details of the PCI bus architecture are described in the publication "PCI Local Bus Specification," Revisions 2.2 (Dec. 1998), published by the PCI Special Interest Group, which publication is incorporated herein by reference in its entirety. Each PCI device, also referred to as a bus master or target, that communicates on the PCI bus includes a configuration space including information used to address the device on the PCI bus.

[0003] The PCI specification provides for delayed transaction processing. In delayed transactions, the master submitting a read request is disconnected from the bus while the target device accesses and buffers the requested data. The master initiating the read would continually retry the read until the requested data is gathered in the buffer. When the master retries the read request after the target has gathered the requested data, the data will then be returned to the master. In this way, the bus is not held in wait states while the read data is gathered and other devices can access the PCI bus. This is especially important for read requests directed toward slower target devices where the read operation can take longer to complete. In such cases, the delayed read request will avoid the lengthy read operation from occupying the bus and preventing other devices access during the lengthy read.

[0004] When multiple masters are connected to the PCI bus, other master requests may take control of the PCI bus for extended periods and prevent the master that was

disconnected for the delayed read transaction from accessing the requested read data from the buffer for the extended period. For instance, if after a master disconnects as part of a delayed read transaction, another master can submit a posted write. The master that submitted the read may not reconnect and retrieve the data until the intervening write request has completed. Such delays due to an intervening write can add significant delays to the read request, especially for lengthier writes. A posted write is a write where upon transferring the data to an intermediate agent, such as the target in a PCI device or bridge, the transaction completes at the originating agent before it completes at the intended destination, e.g., the data is written to the target device. This allows the originating agent to proceed with the next transaction while the requested transaction is working its way to the ultimate destination.

[0005] In the prior art, the read master can only reconnect after the master write completes the transfer of data across the bus or after a latency timer expires, which effectively places an upper limit on any master's access. After the posted write has completed on the PCI bus or the latency timer has expired, the write master is disconnected and the read master can reconnect to access the buffered data. Such delays to the read request can be extensive if the write request is lengthy. Moreover, the latency timer is often set to a value much longer than the time required for the target to fetch the requested read data, thus causing latency from the time the data is available to when the latency timer expires.

[0006] One goal of PCI bus designers is to minimize read latency delays. Read latency is a result of both the delays that occur when accessing the data from the target device and any delays resulting from other masters gaining control of the PCI bus and preventing the delay read master from reconnecting, which occurs in the case of a posted write. If the system performance is particularly sensitive to read latency delays, then the frequent occurrence of such delays may degrade system performance. For these reasons, there is a need in the art to reduce read latency delays that occur when intervening requests, such as posted writes, prevent the master initiating the delayed read transaction from reconnecting to the bus to access the buffered read request data.

SUMMARY OF THE PREFERRED EMBODIMENTS

[0007] Provided are a method, system, and program for processing operations in a system including a bus, a target device and devices capable of accessing the target device over the bus. The target device receives a transaction request from one of the devices over the bus and determines whether a delayed read request is pending after receiving the transaction request. The target device issues a command to disconnect the device initiating the transaction request from the bus. The device initiating the transaction request is allowed to reconnect to the bus and complete the transaction request after the delayed read request is completed.

[0008] In further implementations, the command to disconnect comprises a retry disconnect that occurs before data subject to the transaction request is transmitted.

[0009] Still further, a determination may be made of whether requested data for the delayed read request is accessed and available to return. In such cases, the command to disconnect the device initiating the transaction request is issued after the requested data for the delayed read request is determined to be available to return.

[0010] Moreover, the transaction request may be allowed to proceed if the delayed read request is pending and if the requested data for the delayed read request is not available to return.

[0011] In still further implementations, a determination is made as to whether a variable indicates a first state or a second state, wherein the state indicated by the variable determines when the target device issues the command to disconnect the device initiating the transaction request while the delayed read request is pending.

[0012] Still further, the command to disconnect the device initiating the transaction request is issued when the device attempts to connect to the target device if the variable indicates the first state. The command to disconnect the device initiating the transaction request is issued after all the requested data for the delayed read request is determined to be available to return if the variable indicates the second state.

[0013] Described implementations provide techniques for handling transaction requests to a target device that occur while a delayed read request is pending at the target device. With the described implementations, the target device will disconnect from the device initiating the transaction request so that the bus remains available for

the delayed read request to reconnect and access the requested data. Such a system minimizes the delayed read latency by preventing another transaction request from blocking the delayed read request from reconnecting when the requested data is available to return.

BRIEF DESCRIPTION OF THE FIGURES

[0014] Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates a PCI bus architecture in which aspects of the invention are implemented; and

FIGs. 2 and 3 illustrate logic to process transaction requests in accordance with implementations of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0015] In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several embodiments of the present invention. It is understood that other embodiments may be utilized and structural and operational changes may be made without departing from the scope of the present invention.

[0016] FIG. 1 illustrates a PCI bus device 2 in which aspects of the invention are implemented. PCI master devices 4a and 4b can communicate on the PCI bus 6 to initiate requests to any of the memory regions within the PCI bus device 2 or any further downstream targets and receive requests from other devices. Although only two master devices 4a, 4b are shown, there may be additional master devices connected to the PCI bus 6, as well as other non-master bus devices. For instance, the PCI bus device 2 may be part of a PCI-to-PCI bridge where other devices connected to another PCI bus device (not shown) can communicate requests to the PCI bus device 2 over another PCI local bus (not shown) directed to one of the masters 4a, 4b.

[0017] The PCI bus device 2 includes initiator 8 hardware to send read and write requests to other PCI devices, such as masters 4a, 4b or any other PCI device over the

PCI bus 6. The PCI bus device target 10 comprises hardware to receive read and write requests asserted on the PCI bus 6 that are addressed toward the address space of the PCI device 2.

[0018] The PCI bus device 2 is configured to have target base addresses that define addressable locations in memory components accessible through the target 10, including a Static Random Access Memory (SRAM) 12, internal registers 14, and Synchronous Dynamic Random Access Memory (SDRAM) 16. In certain implementations, the target 10 communicates with the memory components 12, 14, 16 via an internal bus 17. The internal registers 14 include a delayed read response mode variable 18 that indicates a state of how delayed transactions are handled by the target 10. The state value for the variable 18 may be set during initialization or in response to a user entered specified state command that occurs as part of configuration or after configuration during operations of the PCI bus device 2. The registers 14 may include the configuration space for the target 10. The target 10 would buffer requested data that is accessed from the memory components 12, 14, and 16 in data buffer 20 to return to requesting devices, such as the master devices 4a, 4b. For delayed read transactions, after disconnecting, the target 10 would buffer the accessed read data in the data buffer 20. When the master 4a, 4b or other bus device initiating the read later reconnects to the target 10, the target 10 would then return the data from the data buffer 20 to complete the read transaction.

[0019] FIG. 2 illustrates logic implemented in the target 10 to process delayed read transactions. Control begins at block 50 upon receiving a delayed read transaction from a master 4a, 4b, referred to as the "read master". Certain regions of the memory components 12, 14, and 16 that have particularly long access times, such as the SDRAM 16, are designated as delayed read regions. All read requests directed to a delayed read region will be processed as a delayed read transaction to disconnect the read master from the PCI bus 6. The purpose is to prevent transactions which have longer latency from occupying the PCI bus 6 and preventing other devices from accessing the bus 6. Other memory address regions in the target 10 may not be designated as delayed read regions. The target 10 can concurrently process read/write requests to different regions. The target 10 will latch (at block 52) the

address of the read master and queue the read request in a read queue (not shown). The target 10 then issues (at block 54) a retry disconnect to the read master and proceeds to fetch (at block 56) the requested data into the data buffer 20.

[0020] Upon receiving (at block 70) a retry from the read master, if (at block 72) the requested data is not yet in the data buffer 20, then the target 10 returns (at block 74) a retry disconnect to the read master. Otherwise, if the requested data is available, then target 10 returns (at block 76) the requested data from the data buffer 20 via the PCI bus 6 and removes (at bock 78) the entry in the read queue for the delayed read transaction.

[0021] FIG. 3 illustrates logic implemented in the target 10 to handle an intervening transaction, such as a posted write, non-posted write, read, etc., that occurs during a pending delayed read transaction depending in accordance with implementations of the invention. The operations the target 10 performs are based on the delayed read response mode variable 18 value, which may indicate one of two states, where each state specifies a particular mode for handling intervening transactions during the pendency of a delayed read transaction. Control begins at block 80 where the target 10 receives a transaction request from another device. This transaction may comprise a read or write request, such as a posted write request. If (at block 82) there is no pending delayed read transaction, then the target 10 signals (at block 84) the master 4a, 4b initiating the transaction, which may comprise any PCI transaction (e.g., a posted write, non-posted write, read request, etc.) to continue with the transaction. If the transaction is a new transaction, then the target 10 would continue the transaction by performing whatever operations beginning the transaction entails, receiving writes, reading data from memory components 12, 14, 16, forwarding the request to a secondary PCI bus, etc. If the transaction is a reconnect of a previously disconnected transaction, then continuing the transaction may comprise returning buffered data to a delayed read request or otherwise continuing with a previously discontinued transaction. Otherwise, if (at block 82) there is a delayed read transaction in the read queue and if (at block 86) the delayed read response mode variable 8 indicates the first state, which may correspond to a value of zero as shown in FIG. 2, then the target 10 returns (at block 88) a retry disconnect to the intervening transaction. The

retry disconnect causes the intervening master to continually retry the read/write request. After the delayed read has completed, then the disconnected transaction may reconnect to perform the intervening transaction.

[0022] With the first state technique, read latency is reduced because the intervening transaction will not hold the PCI bus 6 and prevent the delayed read from immediately accessing the requested data once the data is available in the data buffer 20. The target 10 leaves the PCI bus 6 open for the delayed read to reconnect and retrieve the requested read data from the data buffer 20. Further, because in most cases, the time needed to retrieve the data from the data buffer 20 is substantially shorter than the time needed to complete the intervening transaction, such as receiving write data from a posted or non-posted write, accessing and transmitting data for a read to a memory region that is not a delayed read region, i.e., a faster access memory region, any delays to the intervening transaction are offset by improvements to processing the delayed read. In fact, making the intervening transaction wait for the delayed read to gather the requested data from the data buffer 20 may result in a delay that is significantly shorter than making the delayed read wait to for the intervening transaction to complete by transmitting data over the PCI bus 6 or accessing and reading data for the intervening transaction.

[0023] If (at block 86), the delayed read response mode 18 was for a second state handling, which may correspond to a value of one, then the target 10 would first check (at block 90) if all the requested delayed read data is in the data buffer 20. If so, then the target 10 would return a retry disconnect to the intervening master in order to leave the PCI bus 6 open so that the delayed read master 4a, 4b may immediately retrieve the requested data that is available in the data buffer 20. Otherwise, if (at block 90) all the requested delayed read data is not available in the data buffer 20, then the target 10 would continue processing (at block 92) the intervening transaction. At the point at block 92, the intervening transaction can perform read/write on the PCI bus 6 without adversely affecting the delayed read latency because the read master 4a, 4b would not be able to gather the requested data because the requested data is not available in the data buffer 20.

[0024] From block 92, one of two separate events may occur while the target 10 is processing the intervening transaction, the intervening transaction completes (at block 96) or the data buffer 20 becomes filled (at block 100) with all the requested delayed read data. At block 96, the intervening transaction completes and, in response, the target 10 removes (at block 98) the intervening transaction from a transaction queue, such as a read or write queue (not shown). At block 100, all the requested data for the delayed read transaction is added to the data buffer 20. In response, if (at block 102), another intervening transaction is in progress reading or writing data, then the target 10 issues (at block 104) a target disconnect to the transaction request; otherwise, control ends.

[0025] Issuing the target disconnect at block 104 interrupts the intervening transaction to force the intervening transaction off the PCI bus 6 so that the read master may immediately gather the requested delayed read data in the data buffer 20 the next time the delayed read master retries to reconnect to access the read data. In this way, the delayed read latency is minimized because once the requested data is available, the read master may immediately gather the data without having to wait for the intervening transaction to release the bus 6.

[0026] In certain implementations, if the intervening transaction is another delayed read transaction, then that delayed read transaction will be allowed to gather the requested data in the data buffer 20 and not receive the retry disconnect at block 88 or 104 in order to minimize delayed read latency across all delayed read transactions. In such case, the logic to keep the PCI bus 6 available for a delayed read whose data is available in the data buffer 6 applies only to non-delayed read intervening transactions. In yet further implementations, if data for one delayed read request is in the data buffer 20, the target 10 will only keep the PCI bus 6 open against other delayed read requests that have lower priority, i.e., have not yet been queued or are queued at a lower priority than another delayed read transaction. Thus, for the first state implementation, a delayed read that is retrying to access data available in the data buffer 20 may reconnect only if there is no higher priority delayed read in the read queue. For the second state implementation, a delayed read of lower priority accessing data from the data buffer 20 may be interrupted and disconnected if the

data for a delayed read having higher priority becomes available in the data buffer 20. Alternatively, the delayed read, even of lower priority, may be allowed to complete accessing the requested data without interruption.

[0027] The described implementations provide techniques to minimize the delayed read latency by ensuring that the PCI bus 6 is free for a delayed read transaction to reconnect and collect data gathered in the data buffer 20. Such implementations substantially improve delayed read performance and reduce delayed read latency in situations where a lengthy intervening transaction, such as a long posted write, would otherwise be allowed to remain connected to the bus during the transaction and prevent a delayed read from reconnecting to access requested read data available in the data buffer.

Additional Implementation Details

[0028] The described logic for processing transactions may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The term "article of manufacture" as used herein refers to code or logic implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.) or a computer readable medium, such as a magnetic storage medium (e.g., hard disk drives, floppy disks,, tape, etc.), optical storage (CD-ROMs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, firmware, programmable logic, etc.). Code in the computer readable medium is accessed and executed by a processor. The code in which preferred embodiments are implemented may further be accessible through a transmission media or from a file server over a network. In such cases, the article of manufacture in which the code is implemented may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, infrared signals, etc. Of course, those skilled in the art will recognize that many modifications may be made to this configuration without

departing from the scope of the present invention, and that the article of manufacture may comprise any information bearing medium known in the art.

[0029] In the described implementations, the PCI architecture may include additional PCI devices and bridges other than those shown in FIG. 1.

[0030] The described bus and bus devices implemented the PCI architecture. However, in alternative interface implementations, bus and bridge technology known in the art other than PCI may be used to implement the bridge and bus interfaces.

[0031] In the described implementations, the target issued command such as a retry disconnect or target disconnect to prevent other transactions from occupying the PCI bus. In alternative implementations, different commands may be used to cause the requesting device to disconnect from the bus.

[0032] Certain logic was described as being performed by specific components, such as the target, etc. Notwithstanding, operations described as being implemented within specific components may be implemented elsewhere.

[0033] In the described implementations, the target determined from the delayed read response mode variable 18 (FIG. 1) the operations to perform to prevent another intervening request from occupying the PCI bus. In alternative implementations, the target may simply be coded to perform one of the first or second state operations. In such cases, no read response mode variable is needed.

[0034] In the described implementations, the requested memory regions were located within the PCI bus device. Alternatively, the target memory regions may be external to the PCI bus device.

[0035] In the described implementations, the I/O transaction were directed to memory regions. In alternative implementations, the I/O transactions may comprise a request to any type of Input/Output device known in the art, such as a hard disk drive, tape, printer, display monitor, audio amplifier, etc.

[0036] The preferred logic of FIGs. 2 and 3 describe specific operations occurring in a particular order. In alternative implementations, certain of the logic operations may be performed in a different order, modified or removed. Morever, steps may be added to the above described logic and still conform to the described

implementations. Further, operations described herein may occur sequentially or certain operations may be processed in parallel.

[0037] The foregoing description of the described implementations has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.